Quizz 4 - Rails	Full name
Q1 - How do you create a Rails app?	
1 →	
Q2 - How do you start coding a Rails project? Give the rig1. Coding the views?2. Coding the controllers?3. Coding the models?	ht sequence.
Q3 - How do you generate a song model with a title a	nd a year ?
1 →	
What are the 2 created files?	
What is the rails command you should type then?	
1 ->	
Q4 - How do you add a category (ex: "rock", "electr correct Rails generator?	o", etc) to your songs table using the
1 -	
What is the created file?	
What is the rails command you should type again?	
1 →	
Q5 - Add a validation on the presence of a song title & cra	ash-test your model in the console
1 # models/song.rb	der in the console
2 class Song < ApplicationRecord	
3 # Add the validation 4	
5 end	

Now crash-test your model:

1 → ra	ils c			
2 pry>				
3 pry>				
4 pry>				
5 pry>				
6 pry>				
7 pry>				
8 pry>				
 The Rou The action Everythin The action 	is the Rails flow you need to follow again and again? Give the correct order ter is routing the HTTP request to "controller#action" on is getting data from models ag starts with an HTTP Request on is rendering the view are the 4 different parts inside an HTTP request?			
F				
Q8 - Are the	HTTP requests in the following 2 routes the same? Why?			
1 # con	nfig/routes.rb			
2 get '	/songs" => "songs#index"			
3 post	"/songs" => "songs#create"			
Q9 - What's	the difference between a GET and a POST request?			
b				
Q10 - Comp	plete the controller code using the correct params key?			
HTTP reque	est:			
GET /search?query=thriller				

Routing:

```
1 # config/routes.rb
2 get "/search" => "songs#search"
```

Controller:

```
1 class SongsController < ApplicationController
2  def search
3  # TODO
4  @songs =
5  end
6 end</pre>
```

Q11 - Complete the controller code using the correct params key?

HTTP request:

```
GET /songs/named/thriller
```

Routing:

```
1 # config/routes.rb
2 get "/songs/named/:name" => "songs#search"
```

Controller:

```
1 class SongsController < ApplicationController
2  def search
3  # TODO
4  @songs =
5  end
6 end</pre>
```

Q12 - What are the 7 CRUD routes generated by the resources method in Rails?

```
1 # config/routes.rb
2 # TODO: Give us the details of the 7 routes generated with:
3 resources :songs
4 # HINT: HTTP-verb "url" => "controller#action"
5
6
7
8
9
10
11
```

Q13 - How do you print your routes and their URL prefix helpers?

```
1 →
```

Q14 - How do you generate a controller for your songs?

```
1 →
```

Q15 - Implement the Read actions in your songs controller?

```
1 class SongsController < ApplicationController
2
3
4
5
6
7
8 end</pre>
```

Q16 - What are the 2 requests needed to create a new song? Implement the songs#new and songs#create actions.

```
1 class SongsController < ApplicationController
 2
 3
 4
      end
 5
      def create
 7
 8
 9
10
11
12
13
      end
14
15
     private
16
      def song_params
17
18
      end
19
   end
```

Q17 - Why do we have to filter parameters using "strong params" in the controller?

Q18 - **Hard question**: What is the HTML generated?

```
1 @song = Song.new
```

Now what is the HTML code generated by:

Fill the blanks:

Q19 - Hard question: What is the HTML generated?

Imagine that:

```
1 @song # => <#Song: id: 18, title: "Hey jude", year: 1968, category: "rock">
```

Now what is the HTML code generated by:

Fill the blanks:

Adding a 2nd model

Q20 - Now you want to add reviews to your app. Here are some constraints

- We don't want our visitors to destroy or update reviews, just to create ones.
- We don't want a separate index page to list all reviews or a show page to display each review.
 Instead, we want to display reviews on the show page of each song, for better UX.

Step #1: Model

Generate your Review model in the terminal. It should have only a content:string and a song:references (= the foreign key).

```
1 -
```

Run the migration

```
1 →
```

Add validation/associations

- Add a validation for the presence of a content
- Add associations between Review and Song

```
1 class Song < ApplicationRecord
2
3 end</pre>
```

```
1 class Review < ApplicationRecord
2
3
4 end
```

Step #2: Routing/Controller

Generate the reviews controller

```
1 →
```

Add the **necessary** routes (don't forget **we don't want the 7 CRUD actions for reviews**)

```
1 # config/routes.rb
2 resources :songs do
3 # TODO
4
5 end
```

Now code your controller:

```
1 class ReviewsController < ApplicationController
 2
     before_action :set_song
 3
 4
    def new
 5
 6
    end
 7
 8
     def create
 9
10
11
12
13
     end
14
15
    private
16
    def set_song
17
       @song = Song.find(params[:song_id])
18
    end
19
     def review_params
20
       params.require(:review).permit(:content)
21
     end
22 end
```

Step #3: Views

Add a song's reviews on its show page: